Adaptive highlighting of links to assist surfing on the Internet

Zsolt Palotai*, Bálint Gábor†and András Lőrincz‡

Abstract

The largest source of information is the WWW. Gathering of novel information from this network constitutes a real challenge for artificial intelligence (AI) methods. Large search engines do not offer a satisfactory solution, their indexing cycle is long and creates a time lag of about one month. Moreover, sometimes search engines offer a huge amount of documents, which is hard to constrain and to increase the ratio of relevant information. A novel AI-assisted surfing method, which highlights links during surfing is studied here. The method makes use of (i) 'experts', i.e. pre-trained classifiers, forming the long-term memory of the system, (ii) relative values of experts and value estimation of documents based on recent choices of the users. Value estimation adapts fast and forms the short-term memory of the system. (iii) Neighboring documents are downloaded, their values are estimated and valuable links are highlighted. Efficiency of the idea is tested on an artificially generated sample set, on a downloaded portion of the Internet and in real Internet searches using different models of the user. All experiments show that surfing based filtering can efficiently highlight 10-20% of the documents in about 5 to 10 steps, or less.

Keywords: Internet, search, intelligent crawler, reinforcement learning, adaptivity, information filtering

1 Introduction

The number of documents on the World Wide Web is increasing quickly; it has passed 1 billion in 2001 and might reach 10 billions soon. The number of new documents published on the WWW is much over 1 million per day. The number of documents that change on a daily basis, e.g. documents about news, business, entertainment, etc., is even larger. The ever-increasing growth presents

*Department of Information Systems, Pázmány Péter sétány 1/C, Budapest, Hungary H-1117. Also at the University of Technology and Economics, Pázmány Péter sétány 1/D, Budapest, Hungary H-1117.

†Department of Information Systems, Eötvös Loránd University, Pázmány Péter sétány 1/C, Budapest, Hungary H-1117.

‡Corresponding author. Department of Information Systems, Eötvös Loránd University, Pázmány Péter sétány 1/C, Budapest, Hungary H-1117. E-mail: lorincz@inf.elte.hu.

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED
05 JUN 2003	**N/A**	**-**

4. TITLE AND SUBTITLE	5a. CONTRACT NUMBER
Adaptive highlighting of links to assist sur¯ng on the Internet	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **Department of Information Systems, Pazmany Peter setany 1/C, Budapest, Hungary H-1117.**	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release, distribution unlimited

13. SUPPLEMENTARY NOTES
See also ADM001661.

14. ABSTRACT

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			
unclassified	**unclassified**	**unclassified**	**UU**	**22**	

a considerable challenge in finding, gathering and ordering information on the web. Gathering of novel information is not efficient with conventional search engines. First, the information they present is not up-to-date. Second, these engines may offer hundreds or even thousands of documents, many of which are not really relevant. Other documents could be 'traps', e.g., by making use of particular (sometimes fake) keywords, or being simply collections of documents indexed in every 'dimension' of the web.

Specialized crawlers, possibly intelligent personalized crawlers may circumvent these problems. The development of such crawlers represents a real challenge for methods of artificial intelligence and has been attempted by several research groups [4, 12, 10, 13]. Intelligent crawlers can serve as pre-filtering methods. Our question is how an intelligent crawler could serve human intelligence to search and find information.

Our aim was to establish a connection between human intelligence and artificial intelligence. While the user surfs the web, our system tries to choose the links which are likely the most relevant to him. The level of relevance – the *value* – of a document is estimated by the previous decisions of the user. With this method, the best links can be highlighted in the browser, so as to make the choice between the links easier.

To this end, we developed a set of artificial 'experts' (classifiers) with different methods. Our solution is an alternative to other expert methods, such as the method of mixture of experts and the method of product of experts. Mixture of experts represent a collection (union) of example sets [7]. Product of experts, on the other hand, makes use of multiplicative probability estimation [6].

The method studied here is favored because of its fast adaptation. It has the following main characteristics. We used a set of text classifiers, which had rather sharp decision surfaces. In turn, the output of the classifiers is a vector with values close to $+1$ or -1. Output $+1$ (-1) means that the input belongs (does not belong) to the classifier. In a given decision problem either the positive (within class) or the negative (not within class) outputs of the classifiers could be of importance. It is also possible that neither of these outputs has any relevance to the actual decision making problem. In turn, an expert should have a weight, which can assume positive, zero or negative values and these weights can serve *value estimation* for each document. Value estimation can be improved by reinforcing feedback provided by the user [11]. The decision to highlight a document means that the document is in the best $p\%$ of available documents according to the actual value estimation. The actual value estimation is based on the user's previous choices, whereas percentage p can be set by the user. Classifiers together with their learned weights can be used to estimate the value of visited and novel documents. These estimated values of neighboring documents (to which the links of the actual and the previous documents point to) can be used for highlighting during surfing.

The adaptation algorithm works as follows. The system maintain a weight vector estimation of the user's goal. In every step all the neighboring documents are downloaded, their values are estimated, and – based on which document was selected by the user – the weight vector is updated.

The paper is organized as follows. First, our methods are described in Section 2. This section contains the description of the databases used for testing as well as the description of the adaptation algorithms studied. Section 3 describes the results of the investigations. This section is followed by the discussion on the found properties of different methods (Section 4). Conclusions are drawn in Section 5.

2 Methods

2.1 Value estimation and user modelling

The task can be interpreted as a prediction of the next decision to be made by the user. Two types of questions can be asked during the process. (i) Which is the next document to be chosen by the user? (ii) If we rank the links, how good is this ranking? We shall say that the goodness of our ranking is 90% on average, if on average only 10% of the documents has been given better ranks by the learning algorithm than the document selected the user model. That is, if goodness of ranking is above 90% then about 10% of the documents could be highlighted.

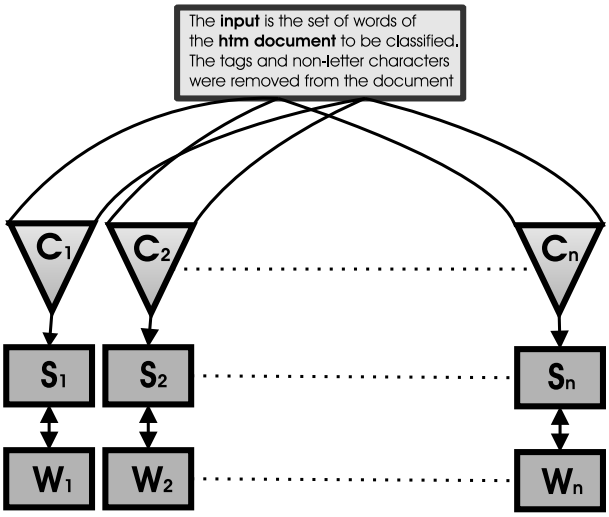

Figure 1: **Value estimation**
There are n different classifiers $(C_i, i = 1, \ldots, n)$. If a new page is downloaded then each classifier provides an output, which is often close to $+1$ or to -1. These real numbers form the classifiers' output vector $\mathbf{S}^T = (S_1, \ldots, S_n)$. The value of each document is estimated by means of the weight vector $\mathbf{W}^T = (W_1, \ldots, W_n)$. The value is given as a scalar product $V(document) = \mathbf{S}^T \mathbf{W}$. Users are modelled by their weight vectors.

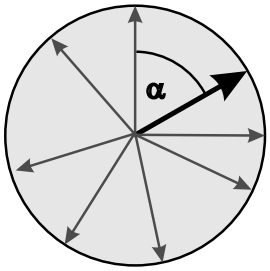

Figure 2: **Model users**
The W and S vectors are normalized to unit length. The thick, black vector represents the 'orientation' of the user. The gray vectors depict the 'orientations' of the neighboring documents. A value of a document is $\cos\alpha$, where α is the angle between the orientation of the user and the document. Note, that the cluster space is of n dimensions ($n = 50$).

To allow fast adaptation we assumed that a surfer follows a – possibly changing – goal and that this goal can be characterized by weighted mixtures of classifiers. Our assumption allows us – in principle – to tolerate a large number of possible goals. To give a rough estimation, let us restrict the choice of weights to ± 1. Then the number of possible goals is 2^{50} for 50 classifiers formed by the 50 clusters. In our experiments we made several restrictions on the possible goals, but the type of restrictions were not known for the learning algorithm.

The outputs of the classifiers were continuous, mostly falling in the neighborhood of -1 and $+1$ values.[1] The output of a classifier can be interpreted as 'reporting' ('expressing opinion') with some uncertainty that a document belongs to the class or not.

We conducted our experiments using user models. These models were simple weight vectors, with several restrictions on their values (see Section 2.4). In some tests the model user could change the topic of interest in order to test the speed of adaptation: a random change of the weight vector was used to test this property.

We asked two questions. The first question was how fast and how precisely the weights of the user model can be estimated during surfing. The second question was whether different classifiers can describe the same goal.

So in our approach, it is assumed that any user can be modelled by a relevance (or weight) vector $\mathbf{W}^T = (W_1, \ldots, W_n)$. For a model user, the value of document S is the scalar product of the output vector ($\mathbf{S}^T = (S_1, \ldots, S_n)$, concatenated from the outputs of the classifiers) and the weight vector, where ^{T} denotes transposition. In turn, the value of document S is estimated by our model as

$$V(S) = \sum_n W_n S_n = \mathbf{W}^T \mathbf{S} \tag{1}$$

[1] On our collection from the *Geocities* database, about 5% of the classifier outputs fell into the domain $(-0.9, +0.9)$, whereas 95% of the outputs were close to either -1 or $+1$.

Figure 1 and 2 depicts value estimation using this weight vector method. The goal of the user identification algorithm is to identify the weights of the user.

2.2 Databases

To study the problem, first, a model user and a model database were created and studied. In the second step, we have downloaded a portion of the internet and tested the highlighting algorithm with a model user on this downloaded portion. The third case concerned highlighting during real Internet searches performed by our user model. In this section databases and our classifying tools are described.

2.2.1 Artificial database generation

A model problem made use of a database defined by an algorithm. Each document was represented by an n-dimensional random number ($n = 50$), whose components chosen from the set of $\{-1, +1\}$, and with an additive superimposed real number chosen from a Gaussian distribution of 0.1 variance. This n-dimensional number was regarded as an output vector of the classifiers for a 'document'. In each trial, we generated data of 100 'documents', 'linked' to the actual 'document'.

2.2.2 The Geocities database

The downloaded portion of the Geocities[2] database contained about 90,000 html documents in 1.5 GB text data. The average branching of one link was about 3 because not all links were downloaded. The maximum branching ratio was about 150, whereas several links had zero branching ratios. This broad distribution resembles to self-evolving networks (such as the web itself), which follow power-law link distribution. Numbers on branching ratio are obtained from 20,000 sample documents chosen randomly.

The structure of part of the downloaded Geocities database shown in Fig. 3 was revealed by a 'breadth first' crawler. This breadth first crawler collected all documents linked to a starting page and then all links of all collected documents and so on. Fig. 3 shows a broad variety of branching ratios, indicating that the Geocities database itself may assume the form typical to *small worlds*. (For details on small worlds, see, e.g., [1] and references therein.)

2.2.3 Tests on the Internet

Highlighting experiments were conducted on the Internet. The starting node was the main page of Geocities. No other restriction was applied; the user could surf the whole web.

[2] `http://www.geocities.com`

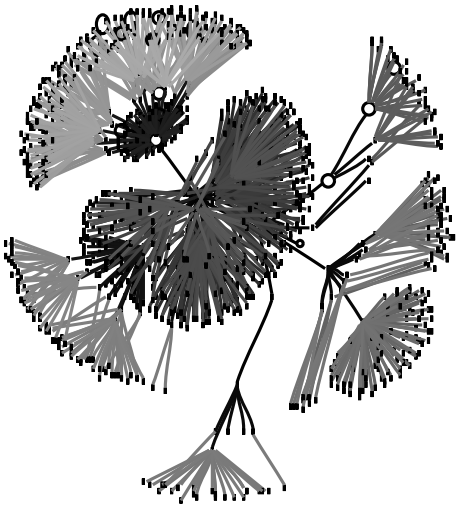

Figure 3: **Structure of the Geocities database**
Relative positions of 10,000 documents found on the downloaded Geocities database using a breadth first algorithm. Circles denote valuable documents containing the word that was searched for. The size of the circles represent the number of valuable documents. (See text for details.)

In the Geocities and the Internet tests we assumed that the user could use the 'back' button or the 'history' utility, so any link seen so far is available at each step.

2.3 Clustering and classification

The downloaded portion of the Geocities database (i.e., 90,000 html documents) were separated into 50 basic clusters by Boley's classification method [2] known to perform well on texts.

For classification of documents in the clusters, the probabilistic term frequency inverse document frequency (PrTFIDF) classification method [8] was used. The term frequency vector had 4,000 components.

2.4 Restrictions on user models and highlighting methods

In all experiments users did not returned to documents which had been visited previously.

2.4.1 The simplest user model on the model database

The user was modelled by a vector with two non-zero components. The values of these components could be either $+1$ or -1. The user chose the 'linked' document with the largest scalar product between the vector of the user and that of the 'linked' document.

2.4.2 User model and highlighting system on the downloaded database

The user was modelled by making use of the Boley classes (see Fig. 2). The user was 'composed' of two of the classifiers with weights either $+1$ or -1 (like in the simplest user model). This vector, which belonged to the convex hull of the classifiers of the highlighting system, had to be identified by the learning algorithm, which used all the 50 Boley classifiers for value estimation.

2.4.3 User models and highlighting systems on the Internet

Three different types of experiments were conducted on the internet, which are depicted in Fig. 5. Each subfigure contains the classifier used to model the user and the classifiers used for highlighting.[3]

In the first type of experiments – from now on it will be referred to as *scheme fullC* (Fig. 5) – the user was defined by one Boley classifier, with weight value 1. The highlighting classifier system contained all the Boley classifiers, like in the previous case.

In the second type of experiments, *scheme C-1*, the same kind of user was assumed (i.e., user was defined by one cluster), but the classifier which represented it was taken off from the set of classifiers in the highlighting system. In turn, value estimation could not use that classifier. The user did not belong to the convex hull of the classifiers and the learning system had to approximate the class which defined the user. This second test concerns the case when our system tries to predict a user 'who' does not match any of our classifiers.

Note that the dimension of the term frequency vector is 4,000. In our tests each classifier (out of the 50 classifiers) was tried as a user. Single value decomposition (see Fig. 4) demonstrates that the frequency vectors of the 50 classifiers are not linear combinations of each other, and therefore, perfect identification is not possible.

In the third type of experiments, the user was not modified. However, moves of the user were predicted by classifiers developed in a different manner: context graph classifiers. A novel and efficient approach in Internet crawling [5, 11, 9] makes use of *context graphs* of relevant documents. Context graphs are constructed as follows: the nodes are documents on the Internet, the oriented edges are links from one site to another. A context graph is a tree, its 0th level (the 'root' level) contains documents from the cluster to which it belongs. Its 1st level contains the sites which have links pointing to a document from the 0th level. Its 2nd level includes sites which have links to a document from the 1st

[3] Note, that the two-classifier user model identification problem is not depicted in the figure.

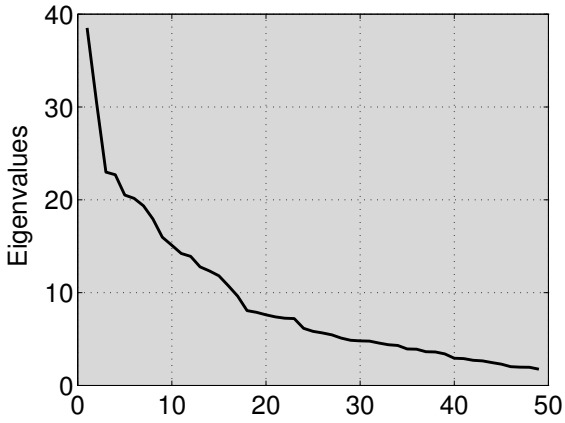

Figure 4: **Single value decomposition of a matrix made of 50 Boley classifiers**
The size of the matrix is 50×4000, it's rank is 50. Taking out 1 of the classifiers the rest can not reconstruct that vector by linear combination of the others.

level, but do not have any link to a 0th level site, and so on. The pseudo-code of constructing the context graph is provided in the Appendix.

Context graphs were developed for each Boley class. The centers of a Boley class (as provided by Boley's method [2]) were chosen as root nodes of the corresponding context graph. Our search for documents to build the context graph was successful for 37 classes; in these cases we found more then 100 documents for 2 CF levels around the Boley clusters. To each of the 37×3 classes, a classifier was built using the cited PrTFIDF classification method. A classifier represented a decision surface between documents of a cluster and the other documents which did not belong to this cluster.

The user was modelled by one of the Boley clusters, and the classifiers belonging to the context graph of this Boley cluster were removed from the classifier vector of the highlighting system. The rest of the PrTFIDF classifiers (i.e, 36×3 classifiers) were used to predict the choices of the model user (scheme CF-3 of Fig. 5). These classifiers were inserted into the value estimation algorithm to be described below.

2.5 Adaptation

2.5.1 On-line reinforcement learning (RL)

RL is the state-of-the-art method in value estimation and it can deal with function approximators.[4] The RL algorithm for learning linear weighting is relatively

[4]For an excellent recent introduction see [15]. Utilization of RL principles in internet surfing can be found in [14, 11].

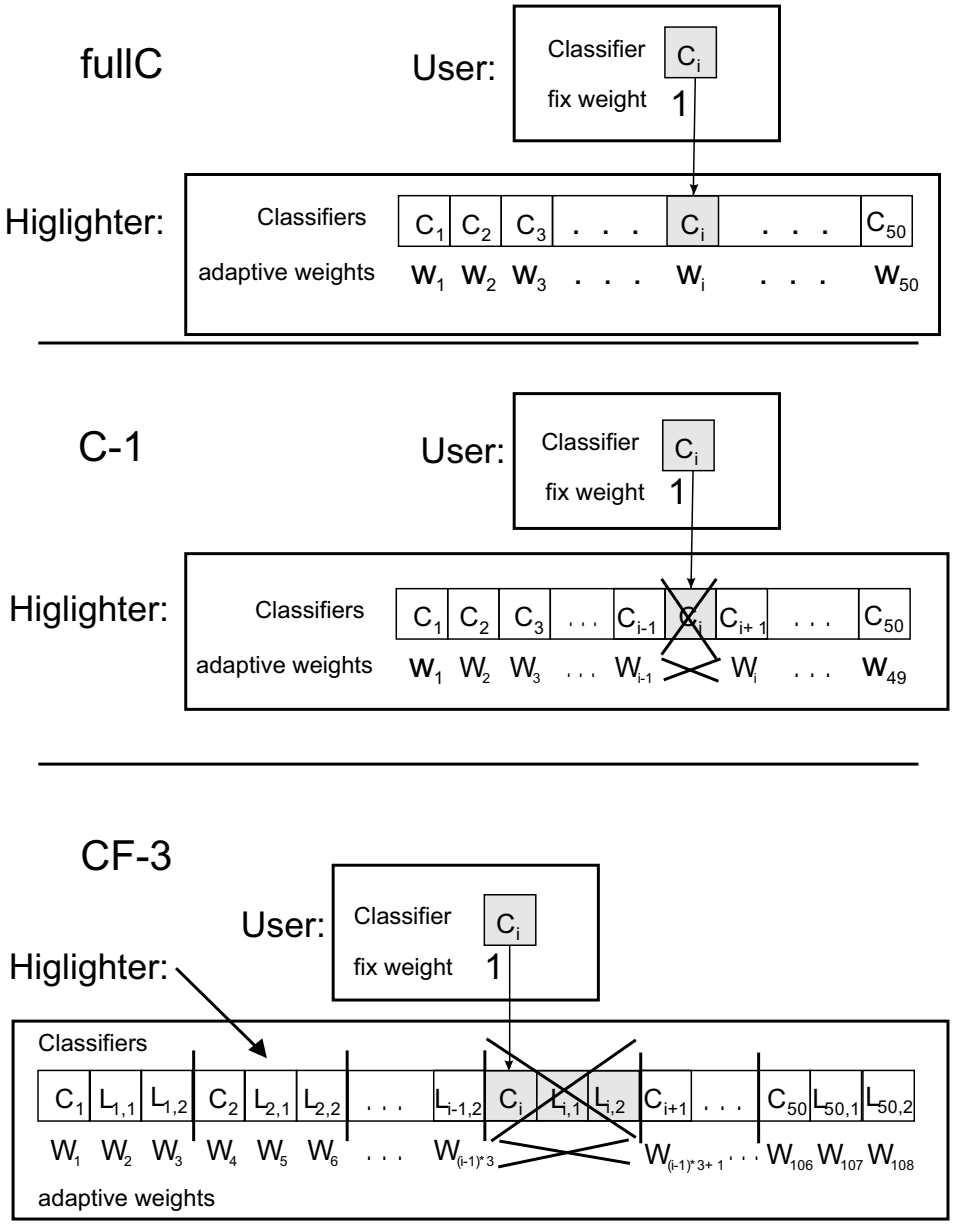

Figure 5: **Schemes of the studies.**
In all cases the user was one of the Boley classifiers. Top (scheme fullC): every Boley classifier takes part in highlighting, middle (scheme C-1): every Boley classifier except the user's classifier takes part in highlighting, bottom (scheme CF-3): every Boley classifier and the belonging context classifiers except those of the user take part in highlighting. Note, that the two-classifier user model identification problem is not depicted in the figure.

simple:

$$\delta(t+1) \;=\; r(t+1) \;+\; \gamma\,\mathbf{W}^T\,\mathbf{S}\,(t+1) \;-\; \mathbf{W}^T\,\mathbf{S}(t) \tag{2}$$

$$\mathbf{W}(t+1) \;=\; \mathbf{W}(t) \;+\; \alpha\,\delta\mathbf{S}(t+1) \tag{3}$$

where

$$V(S(t)) \;=\; \mathbf{W}^T(t)\,\mathbf{S}(t)$$

$V(s) \in \mathbb{R}$ is the estimated value of documents, $\mathbf{W}(t) = (W_1(t), \ldots, W_n(t))$ is the estimated weight vector (or preference vector) of the user ($n = 50$) at surf step t, $\mathbf{S}(t) = (S_1(t), \ldots, S_n(t))$ is the output vector of the classifiers for the document visited at surf step t, γ is the discount factor, $r(t+1)$ is the immediate reward after the $(t+1)^{th}$ step is made by the user, $\delta(t+1)$ is the error of the value estimation, α is the learning rate, which may depend on time, and Eq. (3) is the update rule of the weight vector. The immediate reward can be determined by the designer of the learning system. In our case the immediate reward was zero if $\mathbf{W}^T(t)\,\mathbf{S}(t+1)$ was the maximum value among all of the neighboring documents. Otherwise, $r(t+1) = -1$ was utilized. Different discount factors, including no discount ($\gamma = 1.0$) were tried. RL warrants improved evaluation of a document visited by the user under certain conditions [15]. Conditions of RL include that the states are known and are Markovian, reward could be stochastic but the distribution function may not change during time. These conditions are not met in web surfing.

2.5.2 Learning with moving window technique

The second method utilized an exponentially weighted update rule:

$$\mathbf{W}(t+1) \;=\; (1-\alpha)\mathbf{W}(t) \;+\; \alpha\mathbf{S}(t+1)$$

Here α was decreased with $1/(\kappa * t)$ and $\kappa = .1$ was chosen. This rule minimizes the mean square error $J = \frac{1}{2}||\mathbf{W} - \mathbf{S}||^2$ according to the Robbins-Monro criterion. The problem with this rule is as follows: weights adapt to the input even when the weights are perfect. In turn, internet regions on different topics may spoil the perfect weight vector.

2.5.3 Learning with value estimation error modulated moving window (VEMW) technique

The third method avoids uncertainties about the learning rate. The advantage of the method is that the learning rate becomes zero automatically if no learning is necessary. This third method is a self-consistent combination of value estimation and the moving window method. The algorithm is as follows:

$$\delta(t+1) \;=\; \mathbf{W}^T(t)\,(\mathbf{S}^*(t+1) \;-\; \mathbf{S}(t+1)) \tag{4}$$

$$\mathbf{W}(t+1) \;=\; \frac{(1-\alpha\delta(t+1))\mathbf{W}(t) + \alpha\delta(t+1)\mathbf{S}(t+1)}{|(1-\alpha\delta(t+1))\mathbf{W}(t) + \alpha\delta(t+1)\mathbf{S}(t+1)|}, \tag{5}$$

where α was set to constant to follow online changes, $\mathbf{S}^*(t+1)$ is the best selection according to the value estimation, $\mathbf{S}(t+1)$ is the selection of the user, and $\delta(t+1)$ is the value estimation error computed at surf step $t+1$. It is easy to see that the estimated value of the user selected link will be increased by Eq. (5), provided that the $\mathbf{S}$ vectors at each step are normalized $|\mathbf{S}(t)| = 1$ for $t = 1, 2, ...$ In this case, $\mathbf{W}(t+1)^T\mathbf{S}(t+1) \geq \mathbf{W}(t)^T\mathbf{S}(t+1)$.

If the user is properly represented by the learned weights, no further weight tuning may occur under this learning rule. Although the algorithm estimates the value, it is not making use of cumulated long-term value estimations, thus it is not an exact RL method. It is more closely related to approximations of the value function like that of STAGE [3]. Connection to RL methods will be provided in Section 4.

3 Results

3.1 Results on artificial 'documents' and 'links'

Computer runs were averaged over 230 independent simulations. In each simulation 200 steps were executed by the model user. Averaged results are shown in Fig. 6. In each computer run we assumed that the user's behavior changes at discrete time steps. For the sake of visual inspection, the occurrences of changes were restricted to the 50th, 100th, and 150th time steps. The figure depicts the average goodness of ranking.

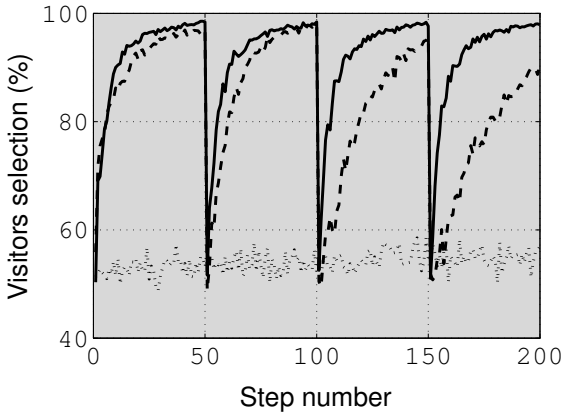

Figure 6: **Comparison of different methods**.
Choice of model user is averaged over 230 generation of 200 link-sets with 100 links in each set. The graph shows the goodness of ranking. The higher the number, the better the highlighting. Solid line: update using VEMW, dashed line: moving window update, dotted line: reinforcement learning. User switched weights randomly in every 50 steps. In the moving window update experiment the learning rate was proportional to $1/t$.

Reinforcement learning performs rather poorly (dotted 'line'). Performance of RL improves rather slowly if there is any improvement at all (Fig. 6). Note, that a goodness value of 50% means that on average, half of documents have better ranking. In turn, random choice from the two options (i) highlight and (ii) do not highlight has a goodness value of 50%.

The moving window estimation is relatively good (dashed line). This method is sensitive to the choice of the learning rate. This can be seen clearly in Fig. 6, where the $1/t$ behavior of the learning rate provides striking deterioration of performance. The moving window method falls below the VEMW method (solid line). Of particular importance is the first 25 steps of our experiments. Within this time domain, the learning rate of the moving window method changes quickly. According to the figure, the initial rise of the moving window method is faster than that of VEMW method. This indicates that the learning rate chosen for the VEMW method is suboptimal. The sharp rise of the moving window saturates and falls below that of the method using value estimation error modulation.

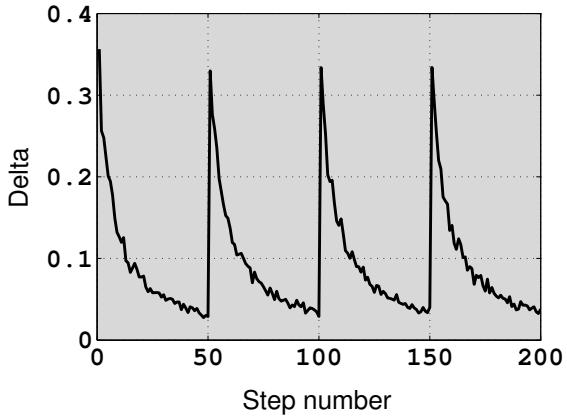

Figure 7: **Value-estimation error as a function of time**.
Changes of the weight vector of the user model occur in every 50th step in the experiment. Value estimation error can be used to monitor changes of the behavior of the user.

One can argue that an optimal choice of the learning rate is not possible. The choice of large rates allows fast adaptation but prohibits fine tuning. On the other hand, small rates allow for fine tuning but may deteriorate performance when changes are fast – as it is demonstrated by Fig. 6. Moreover, there is a pitfall for this moving window estimation: this method modifies the weights in regions where all estimated values are low (i.e.,where all neighboring documents have low values for the user). In such regions highlighting is not relevant. Also, no change of weights is necessary in regions where the error of value estimation is small. The VEMW adaptation (Eq. 4) avoids these problems by construction.

Value-estimation error has other advantages, .e.g., it can serve as an indicator

to measure changes of user's behavior. This is demonstrated in Fig. 7

3.2 Results on the downloaded portion of the Internet

The algorithm had to estimate the user's weight vector, which was changed in every 50th step. Experiments were conducted on the downloaded part of the Geocities database, the adaptation method was the VEMW technique. Averaged results are shown in our figure (Fig. 8). Continuous improvement could be reached only by assuming that links of visited documents are *available* at each instant (Fig. 8). Without this assumption, performance was compromised given the small average branching ratio (about 3) of our Geocities database. This is so, because searches were initiated from random documents. Small branching ratio *and* a wrong starting region together prohibit the estimation of the interest of the user: selections could be meaningless under these conditions. The 'back' button (or the 'history') makes a difference: there is a clear improvement of estimations beyond the 50th and the 100th step. In turn, given the small average branching ratio of the web (around 7), a good highlighting 'agent' provides easy access to the history of surfing.

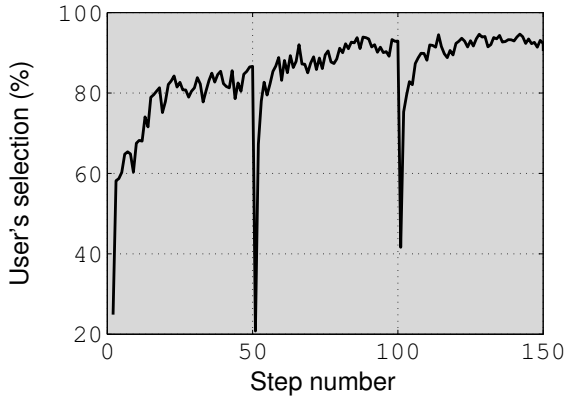

Figure 8: **Results on the Geocities database.**
The simulated user had two nonzero values among the components of vector W (one of the components was set to $+1$, another component was set to -1 and the rest of components were set to zeros). Learning was allowed to tune all components. It was assumed that the 'back' button of the browser is easily available and that possible links are accumulated during surfing. (Note that the average branching ratio of the downloaded part of the Geocities database was about 3.) New weight vector was selected for the 'user' in each 50th step. Results are shown for the update using VEMW.

3.3 Results on the Internet

Four different studies were conducted on the Internet (see Section 2.4.3), each of them used the VEMW technique.

In Study 1 and in Study 2 schemes fullC and C-1 were applied, respectively. Complete tests were made, all classifiers belonging to the Boley clusters were tried as user models in each study. Averaged results (with error bars representing variances) are depicted in Fig. 9(a) and (b). In Study 3, scheme CF-3 was tested. All the 37 classifiers which had CF clusters were tried as users. Averaged results are shown in Fig. 9(c).

Variances are smaller than standard deviations, which – if centered around the average values – would represent a region sometimes beyond 100%. This occurs, because performance, which is good on average, was heavily corrupted from time to time. To show this, performances and changes of the weights are shown in Figs. 10(a) and (b) for the best and for the worst cases of Study 2.

Subfigures of Fig. 9 demonstrate that performances are similar for the three different user types. However, – according to Figs. 10(a) and (b) – performance changes from user to user.

In Study 4 we repeated Study 1, but the user was randomly changed in every 50th steps, to study the speed of the adaptation. Results are shown in Fig. 10(c). Note that the initial transient is relatively long at start, but it becomes much shorter (2 steps) at later changes of the user, as shown by the inset of Fig. 10(c): at the beginning there are not enough good links to choose from.

4 Discussion

4.1 Evaluation of the results

4.1.1 Theoretical considerations

Moving window estimation was relatively good. It is, however, strongly parameter dependent as it is demonstrated in Fig. 6. The learning rate of the VEMW method is suboptimal as it can be seen from the initial rises of the two curves in Fig. 6 between step number 0 and 10. Nevertheless, modulation of the learning rate using the value estimation error helps to overcome the parameter uncertainties of the moving window estimation.

From the theoretical point of view, RL estimates the error of the value based on experienced quantities:

$$\delta(t+1) \;=\; r(t+1) \;+\; V(\mathbf{S}_{experienced}(t+1)) \;-\; V(\mathbf{S}_{experienced}(t)) \qquad (6)$$

where γ was set to one, quantities in the brackets denote step number or time, $\mathbf{S}_{experienced}(t)$ denotes the experienced state (i.e., the actual node) at time t, $V(.)$ denotes the value function, and $r(t+1)$ is the immediate reward received after the step was made at time t. On the other hand, the theoretical formulation of our VEMW update rule (Eq. 4) computes the estimation error as follows:

$$\delta(t) \;=\; V(\mathbf{S}_{experienced}(t+1)) \;-\; V(\mathbf{S}_{expected}(t+1)) \qquad (7)$$

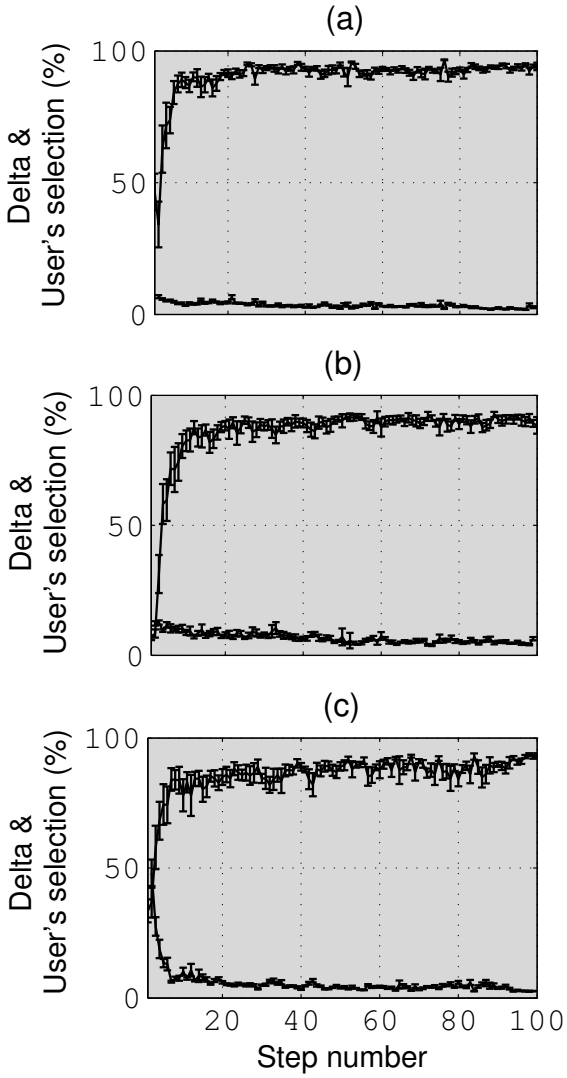

Figure 9: **Performances.** (a): User is from the convex hull of the Boley classifiers, (b): the classifier of the user is excluded from Boley classifiers used for estimation, and (c): the set of classifiers is extended by classifiers belonging to levels of context graphs. Upper curves: performances. lower curves: value estimation errors. Error bars denote variances.

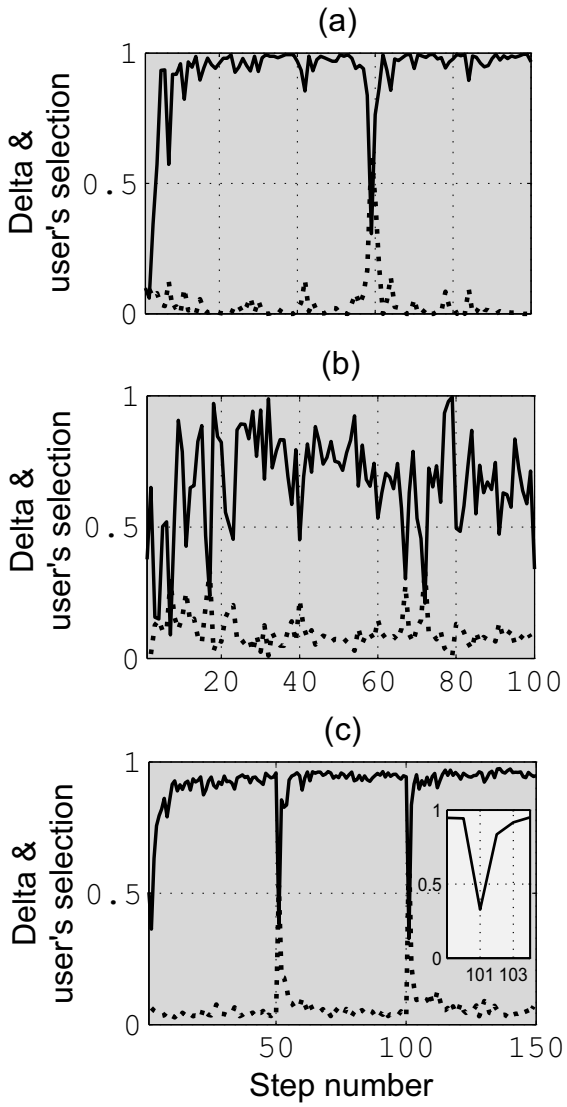

Figure 10: **Individual cases.** (a): The best highlighting result on the Internet for the case (b) of Fig. 9, (b): highlighting with the worst result on the Internet for the same, (c): case (a) of Fig. 9 with a user changing behaviors. Inset: fast adaptation at step number 100.

where subscript 'expected' refers to the state having the best estimated value out of the possible next states. It is intriguing that the learning rule based on this error estimation is much faster than the RL method. However, this is not a real drawback of RL, the VEMW learning can be recast as on RL method. To this end, one needs to replace on-line reward of Eq. (6) with 'expected reward':

$$r_{expected}(t+1) \; = \; V(\mathbf{S}_{experienced}(t)) \; - \; V(\mathbf{S}_{expected}(t+1)) \tag{8}$$

In this case, Eq. (2(and Eq. (7) become identical. $V(\mathbf{S}_{expected}(t+1))$ is available in the highlighting problem. However, it is not available in most RL methods. It is available only if planning, or evaluation of future states are possible. This is the case of finite state games, e.g., chess or backgammon.

4.1.2 Highlighting using Boley clustering

Performance was similar for the users constructed from Boley classifier either when all Boley classifiers, the other Boley classifiers and the other Boley classifiers together with many additional CF classifiers were available. In case when the approximation had the potential to fully identify the user (i.e., when linear combination of Boley classifiers were approximated by *all* Boley classifiers) performance was somewhat better. Few notes should be made here.

The increase of the number of classifiers did not improve performance. Moreover, the weights and the changes of the weights were similar for the 50 Boley classifiers and for the case when CF classifiers were included (Fig. 11). In turn, the set of 50 Boley classifiers seems a reasonable choice.

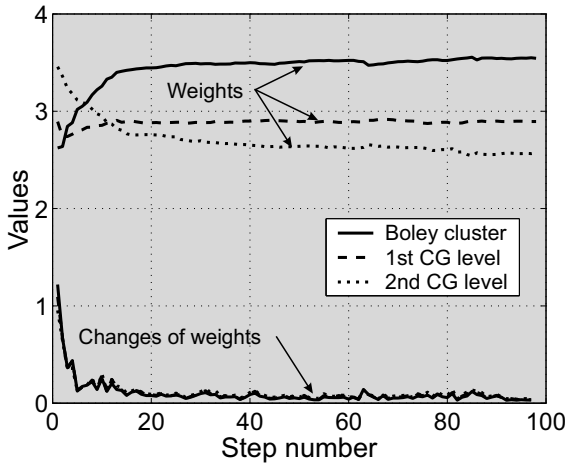

Figure 11: **Weights and changes of weights**
The sum of the absolute values of the weights (upper curves) and the sum of the absolute values of the changes of the weights are depicted for the three different experiments. (CG: context graph)

Performance was never perfect, although it could have been, at least for the case of scheme fullC. This may be due to noise and/or to changing local minima in parameter space, which may arise in a particular domain of documents. Part of the problem originated from the learning rule itself as it is discussed below.

Performance could be improved considerably by modifying the learning rule. The point is that drastic changes in the weight vector may occur when there is no good document at all and the ranking of both the user and the highlighting system is, in effect, random. Large estimation errors may arise, but if the preferences of the user are steady then the weight vector should not be modified. However, similar situation will arise if the behavior of the user changes. The value estimation error will be large, and in this case the weight vector should be modified. These are contradictory requirements for a learning system. One possible solution to this dilemma is to maintain two parallel estimations. The first estimation, which is identical to the one we used, could work in the background. The visible estimation would neglect changes in poor (low value) regions. The learning algorithm could 'switch' between the two estimations based on their predictive powers. In other words, if large changes in the weight vector occurs, then learning is granted only if performance is improved.

Another point concerns the large differences between different Boley clusters. For a uniform and improved performance the contents of these clusters need to be uncovered. We have noticed that there is a strong topic specificity whether a given user model can followed or not. This issue requires further studies. Pre-clustering of users could be a solution here.

4.2 Differences among highlighting, monitoring and crawling

Highlighting can help to review an internet site and its immediate environment quickly. As such, it can decrease search time. Moreover, it may decrease traffic at portals or gateways and could serve as the basis of dynamical home pages. Highlighting can be very efficient but may miss information. Highlighting requires fast adaptation because it serves the user during an exploration. During surfing, both the topic of interest of the user and place of valuable information could be unknown.

Monitoring is different. Monitoring assumes that the place of information is known. It pays constant attention to a given site or a particular set of links. Monitoring collects recently published ('breaking news' type) information. In turn, monitoring will not miss information at the known places. However, monitoring may increase internet traffic and may miss information which appear at new locations. Monitoring may not require adaptation.

Intelligent *crawling* is somewhere in between highlighting and monitoring. In this case, the topic of the search is more or less known, whereas the place of novel information is considered unknown. Adaptation is necessary under this condition, because different regions of the internet require different evaluations, as it has been demonstrated in the literature. [11, 9].

Highlighting, crawling and monitoring are all important ingredients for finding novel information. On the other hand, such tools are not to replace but to complement classical browsers, search tools and databases. The importance of these novel tools is striking when fresh information is searched for. Combination of search engines, crawlers, site monitors and highlighting techniques are all needed for efficient fast information access.

For example an intelligent crawler may serve the user better if it performs highlighting when the user is surfing and performs reinforcement based document collection when the user is absent. The working of an intelligent crawler equipped with highlighting option is depicted in Fig. 12.: One might say that the list of hits and misses used to reinforce the crawler is closely related to bookmarks and making notes about misleading information, respectively. If reinforcement is given carefully, the crawler can collect relevant documents during the absence of the user and the user may have more time for explorative surfing. The presence of the explorative user, who may change his/her behavior frequently, opens the demand for interactive highlighting to assist exploration.

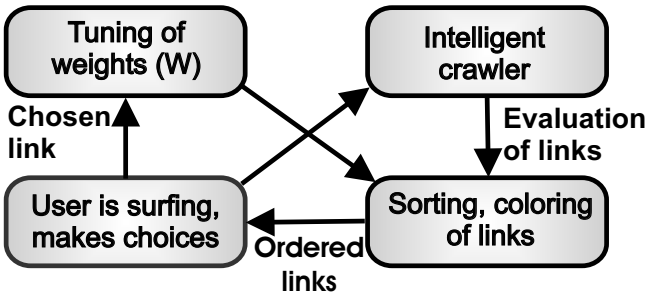

Figure 12: **Scheme of interaction amongst user, intelligent crawler and highlighting systems**

After the user selected his/her step one can estimate or update his/her weight (relevance) vector (W). With this new estimation, documents neighboring the actual link can be analyzed and ranked (ordered, sorted and color-coded). Ranking can be offered directly (via a list), or indirectly, via coloring of links. An intelligent crawler can 'travel' over the internet and can collect possibly relevant documents. Collected documents can be inserted into a list of *offered* documents. Any good (bad) document can be sent to the intelligent crawler for positive (negative) reinforcement to improve crawling.

5 Conclusions

Users surfing on the Internet can be assisted in various ways. Intelligent crawlers can perform pre-filtering. On-line assistance, however, requires fast adaptation of such pre-filtering methods. In this paper a pre-trained, modifiable and extendable classification scheme was suggested for on-line guidance using exploration

of the neighborhood of user-selected documents. It was shown that on-line reinforcement learning performs poorly relative to moving window methods. Value estimation error modulated moving window, which is closely related to reinforcement learning, however, showed improved efficiency. Both the artificial and for the Internet studies, highlighting could filter out about 80% of the documents in less than 10 steps. Sometimes, this number was as low as 2. Future research is needed to discover the 'human factor', when the highlighting agent interacts with real users.

6 Acknowledgments

This material is based upon work supported by the European Office of Aerospace Research and Development, Air Force Office of Scientific Research, Air Force Research Laboratory, under Contract No. F61775-00-WE065. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the European Office of Aerospace Research and Development, Air Force Office of Scientific Research, Air Force Research Laboratory. Careful reading of the manuscript is gratefully acknowledged to Attila Meretei.

7 Appendix

Pseudo-code for context graph generation

```
Building Context Graph
pseudo-code
1. Initialization
   root docs = documents in the cluster
   new empty context graph (CG)
   branching ratio ← n (n = 2.5)
   min number of root docs := m (m = 100)
   0th level of CG := 0th level of CG + root docs
   sort root docs by its distances from the cluster center
2. Constructing the Context Graph
   for the first 200 root docs:
      if CG is Ready then
         remove documents from the 0th level of CG which doesn't linked by any
            document in the 1st level of CG
         CG is ready, STOP
      page := next doc in root docs
      Search For Backward Links To Level 0(page)
      for all new d documents in the 1st level of CG:
         Search For Backward Links To Level 1(d)
3. Checking the size of Context Graph
   if number of documents in each level of the CG > 100 then
      remove documents from the 0th level of CG which doesn't linked by any
        document in the 1st level of CG
      CG is ready, STOP
   else Building of CG failed
end Building Context Graph
```

```
function CG is Ready
   return (size of the 0th level of CG >= min number of root docs and
           size of the 1st level of CG >= (size of the 0th level of CG) * branching ratio and
           size of the 2nd level of CG >= (size of the 0th level of CG) * (branching ratio)^2)
end function

proc Search For Backward Links(level, doc)
   for each l link in doc:
      if level=0 and there is a link on l to doc then
         Add To Context Graph(l)
      if level=1 and
            there is a link on l to any document in the 1st level of CG then
         Add To Context Graph(l)
end proc

proc Add To Context Graph(doc)
   for each level i 1 to 2:
      if there is a link in doc to a document at the (i-1)th level and
      the number of documents pointing to that document < 2*branching ratio and
      the average number of documents pointing to a document < branching ratio and
      the CG doesn't contain doc at smaller levels then
         add doc to the ith level of CG
end proc
```

References

[1] R. Albert and A.L. Barabási, *Statistical mechanics of complex networks*, Reviews of Modern Physics **74** (2002), 47–91.

[2] D.L. Boley, *Principal direction division partitioning*, Data Mining and Knowledge Discovery **2** (1998), 325–344.

[3] J. A. Boyan and A. W. Moore, *Learning evaluation functions to improve optimization by local search*, Journal of Machine Learning Research **1** (2000), 77–122.

[4] S. Chakrabarti, D.A. Gibson, and K.S. McCurley, *Surfing the Web backwards*, 8th World Wide Web Conference (Toronto, Canada), 1999, `http://www8.org/w8-papers/5b-hypertext-media/surfing/`.

[5] M. Diligenti, F. Coetzee, S. Lawrence, C. Lee Giles, and M. Gori, *Focused crawling using context graphs*, 26th International Conference on Very Large Databases, VLDB 2000 (Cairo, Egypt), 10–14 September 2000, `http://www.neci.nec.com/ lawrence/papers/focus-vldb00/ focus-vldb00.ps.gz`.

[6] G.E. Hinton, *Training products of experts by minimizing contrastive divergence*, Technical Report TR-2000-004, Gatsby Computational Neuroscience Unit, University College London, 2000, `http://www.gatsby.ucl.ac.uk/`.

[7] G.E. Hinton, B. Sallans, and Z. Ghahramani, *Learning in graphical models, Ed.: M.I. Jordan*, ch. Hierarchical community of experts, pp. 479–494, MIT Press, Cambridge, MA, 1999.

[8] T. Joachims, *A probabilistic analysis of the Rocchio Algorithm with TFIDF for text categorization*, Proc. Int. Conf. on Machine Learning (ICML), Morgan Kaufmann, San Francisco, 1997, `http://www.cs.cornell.edu/People/ tj/publications/`.

[9] I. Kókai and A. Lőrincz, *Fast adapting value estimation based hybrid architecture for searching the world-wide web*, Applied Soft Computing **2** (2002), 11–23.

[10] S. Lawrence, *Context in web search*, IEEE Data Engineering Bulletin **23** (2000), 25–32.

[11] A. Lőrincz, I. Kókai, and A. Meretei, *Intelligent high-performance crawlers used to reveal topic-specific structure of the WWW*, Int. J. Founds. Comp. Sci. (2002), 477–495.

[12] A. McCallum, K. Nigam, J. Rennie, and K. Seymore, *Building domain-specific search engines with machine learning techniques*, AAAI-99 Spring Symposium on Intelligent Agents in Cyberspace, 1999, `http://www.cs.cmu.edu/~mccallum/papers/cora-aaaiss98.ps`.

[13] S. Mukherjea, *WTMS: A system for collecting and analyzing topic-specific web information*, 9th World Wide Web Conference, 2000, `http://www9.org/w9cdrom/293/293.html`.

[14] J. Rennie, K. Nigam, and A. McCallum, *Using reinforcement learning to spider the web efficiently*, Proc. 16th Int. Conf. on Machine Learning (ICML), Morgan Kaufmann, San Francisco, 1999, pp. 335–343.

[15] R. Sutton and A. G. Barto, *Reinforcement learning: An introduction*, MIT Press, Cambridge, 1998.